**Exercise 1: Configuring a Basic Spring Application**

**Scenario:**

Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

**Steps:**

1. **Set Up a Spring Project:**
   * Create a Maven project named **LibraryManagement**.
   * Add Spring Core dependencies in the **pom.xml** file.
2. **Configure the Application Context:**
   * Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
   * Define beans for **BookService** and **BookRepository** in the XML file.
3. **Define Service and Repository Classes:**
   * Create a package **com.library.service** and add a class **BookService**.
   * Create a package **com.library.repository** and add a class **BookRepository**.
4. **Run the Application:**

Create a main class to load the Spring context and test the configuration

Pox.xml file:

<project

xmlns="https://maven.apache.org/POM/4.0.0"

xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="https://maven.apache.org/POM/4.0.0

https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.24</version> <!-- specify real version number -->

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>5.3.39</version>

</dependency>

</dependencies>

</project>

BookRespirotory.java:

**package** com.library.repository;

**import** com.library.model.Book;

**public** **class** BookRepository {

**public** **void** save(Book book) { /\*...\*/ }

// other data access methods

}

BookService.java

**package** com.library.service;

**import** com.library.model.Book;

**import** com.library.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository repository;

**public** **void** setRepository(BookRepository repository) {

**this**.repository = repository;

}

**public** **void** addBook(Book book) { repository.save(book); }

}

Book.java

**package** com.library.model;

**public** **class** Book {

**private** String title;

**public** Book() { }

**public** Book(String title) {

**this**.title = title;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

@Override

**public** String toString() {

**return** "Book{title='" + title + "'}";

}

}

App.java

**package** com.library;

**import** com.library.model.Book;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** com.library.service.BookService;

**public** **class** App {

**public** **static** **void** main(String[] args) {

ClassPathXmlApplicationContext ctx =

**new** ClassPathXmlApplicationContext("spring/applicationContext.xml");

BookService service = (BookService) ctx.getBean("bookService");

service.addBook(**new** Book("1984"));

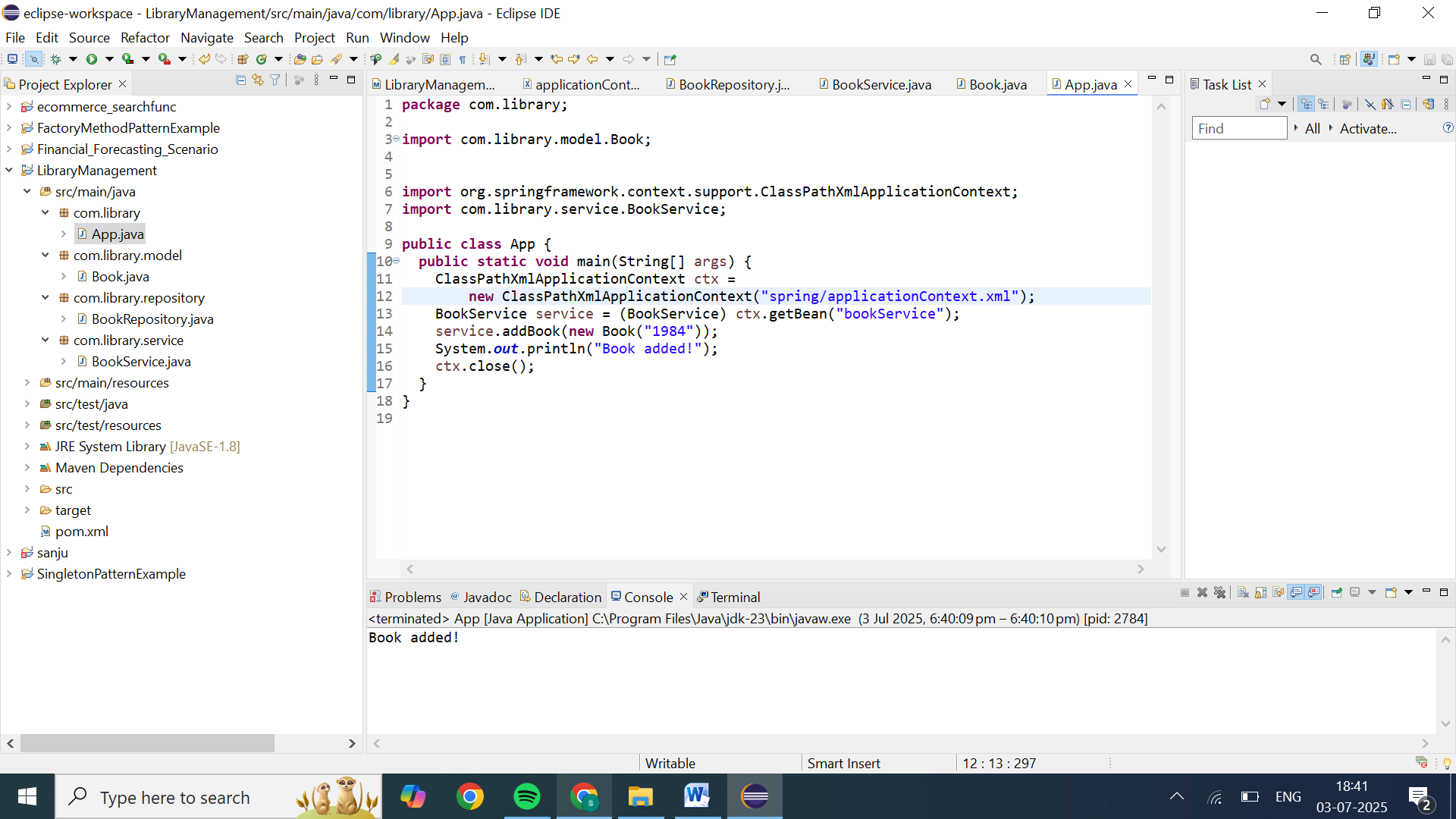
System.***out***.println("Book added!");

ctx.close();

}

}

Output:



**Exercise 2: Implementing Dependency Injection**

**Scenario:**

In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

**Steps:**

1. **Modify the XML Configuration:**
   * Update **applicationContext.xml** to wire **BookRepository** into **BookService**.
2. **Update the BookService Class:**
   * Ensure that **BookService** class has a setter method for **BookRepository**.
3. **Test the Configuration:**
   * Run the **LibraryManagementApplication** main class to verify the dependency injection.

**applicationContext.xml:**

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="bookRepository" class="com.yourorg.repository.BookRepository"/>

<bean id="bookService" class="com.yourorg.service.BookService">

<property name="bookRepository" ref="bookRepository"/>

</bean>

</beans>

**BookRepository.java:**

**package** com.yourorg.repository;

**public** **class** BookRepository {

}

**BookService.java:**

**package** com.yourorg.service;

**import** com.yourorg.repository.BookRepository;

**public** **class** BookService {

**private** BookRepository bookRepository;

**public** **void** setBookRepository(BookRepository bookRepository) {

**this**.bookRepository = bookRepository;

}

**public** BookRepository getBookRepository() {

**return** bookRepository;

}

**public** **void** listBooks() {

// Use bookRepository here...

}

}

**LibraryManagementApplication.java:**

**package** com.yourorg;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** com.yourorg.service.BookService;

**public** **class** LibraryManagementApplication {

**public** **static** **void** main(String[] args) {

**try** (ClassPathXmlApplicationContext ctx = **new** ClassPathXmlApplicationContext("applicationContext.xml")) {

BookService svc = ctx.getBean("bookService", BookService.**class**);

**if** (svc.getBookRepository() != **null**) {

System.***out***.println("✅ Repository injected successfully!");

} **else** {

System.***err***.println("❌ Injection failed.");

}

// svc.listBooks();

}

}

}

**pox.xml:**

<project

xmlns="https://maven.apache.org/POM/4.0.0"

xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="https://maven.apache.org/POM/4.0.0

https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.yourorg</groupId>

<artifactId>library-di</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.9</version> <!-- or latest 5.x/6.x -->

</dependency>

<dependencies>

<dependency>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

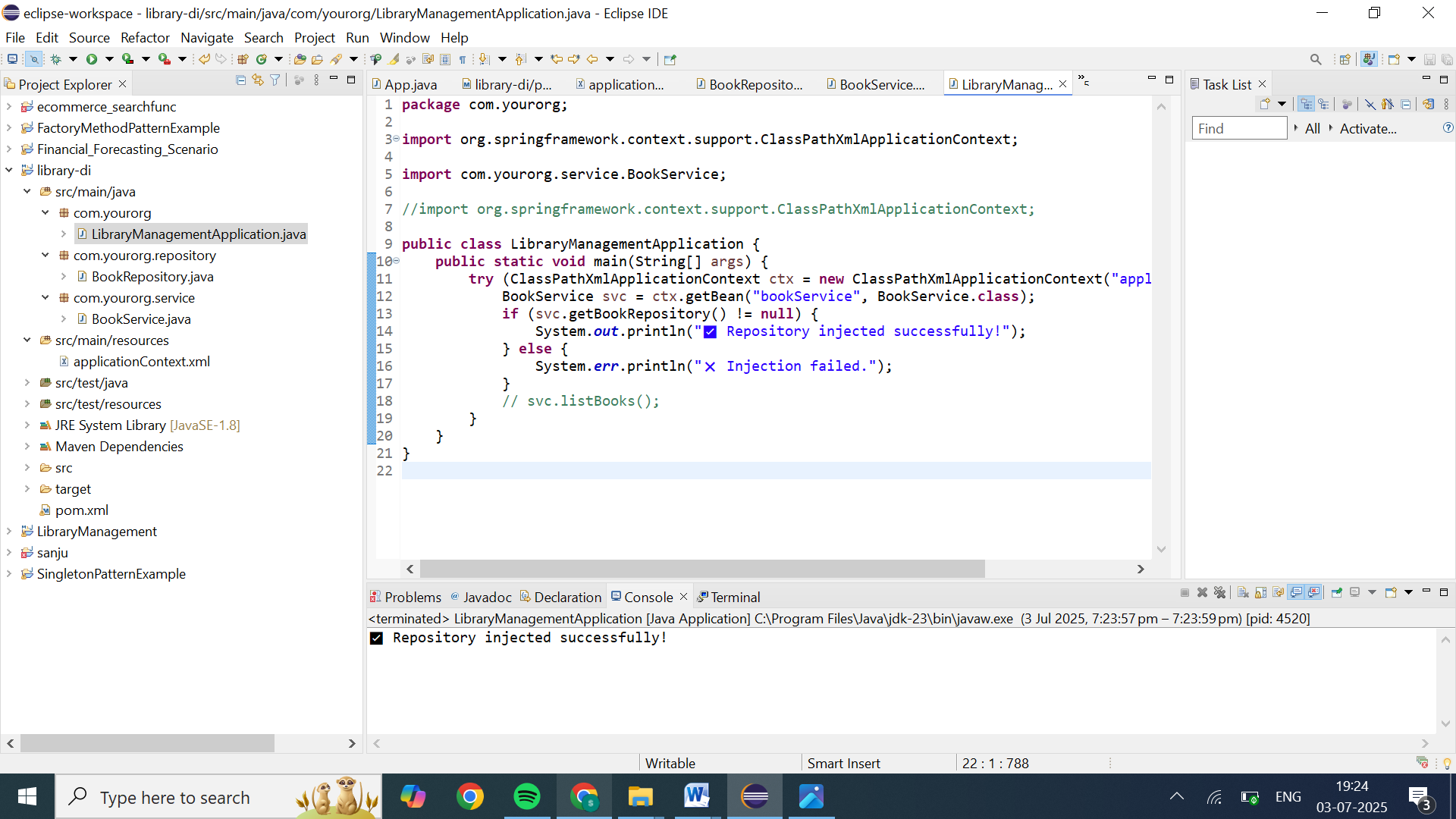
<version>0.0.1-SNAPSHOT</version>

</dependency>

</dependencies>

</project>

Output:



**Exercise 4: Creating and Configuring a Maven Project**

**Scenario:**

You need to set up a new Maven project for the library management application and add Spring dependencies.

**Steps:**

1. **Create a New Maven Project:**
   * Create a new Maven project named **LibraryManagement**.
2. **Add Spring Dependencies in pom.xml:**
   * Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.
3. **Configure Maven Plugins:**
   * Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

**LibraryManagement:**

<project

xmlns="https://maven.apache.org/POM/4.0.0"

xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="https://maven.apache.org/POM/4.0.0

https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>LibraryManagement</artifactId>

<version>0.0.1-SNAPSHOT</version>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.30</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.30</version>

</dependency>

<!-- Spring WebMVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.30</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.0</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**Spring Dependencies:**

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.30</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.30</version>

</dependency>

<!-- Spring WebMVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.30</version>

</dependency>

</dependencies>

**Configure Maven Plugins**

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.0</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

1. Right-click the project → **Properties**.
2. Go to **Java Build Path → Libraries**:
   * Ensure the **JRE System Library** is Java 1.8.
3. Go to **Java Compiler**:
   * Ensure **Compiler compliance level** is set to **1.8**.
   * If not, uncheck “Use compliance from execution environment” and set it to **1.8**

**OUTPUT:**

